

UNCLASSIFIED

## Defense Technical Information Center Compilation Part Notice

ADP010623

TITLE: Development of a Virtual Environment  
Software Testbed Using Commercial-Off-The-Shelf  
Software Components

DISTRIBUTION: Approved for public release, distribution unlimited

This paper is part of the following report:

TITLE: The Capability of Virtual Reality to Meet  
Military Requirements [la Capacite de la realite  
virtuelle a repondre aux besoins militaires]

To order the complete compilation report, use: ADA388966

The component part is provided here to allow users access to individually authored sections of proceedings, annals, symposia, ect. However, the component should be considered within the context of the overall compilation report and not as a stand-alone technical report.

The following component part numbers comprise the compilation report:

ADP010609 thru ADP010633

UNCLASSIFIED

# DEVELOPMENT OF A VIRTUAL ENVIRONMENT SOFTWARE TESTBED USING COMMERCIAL-OFF-THE-SHELF SOFTWARE COMPONENTS

Scott W. Davidson  
Management Systems and Training Technologies Company

Naval Air Warfare Center TSD  
12350 Research Parkway  
Orlando, FL 32826, USA  
Tel: +1 703-528-8015

Corporate Office:  
2111 Wilson Blvd, Suite 1050  
Arlington, VA 22201, USA  
Tel: +1 407-381-8632

scott@rdvax.ntsc.navy.mil

## INTRODUCTION

The goals of the Virtual Environment Training Technology (VETT) program are to develop, demonstrate and evaluate the use of virtual environment (VE) component technologies for navy training applications. A key part of this effort is the development of a testbed, performed by Management Systems and Training Technologies (MSTT), for investigating the benefits of using VE interfaces to improve naval training while reducing training-system costs. This paper describes the technology components integrated by MSTT that provide a tool for training effectiveness research at the Naval Air Warfare Center Training Systems Division (NAWCTSD) in Orlando, Florida. The development approach used for the testbed is to integrate off-the-shelf tools into one cohesive software platform. Applications developed on the testbed include a multi-modal Virtual Electronic Systems Trainer, a spatialized audio location task, and relative motion experiment for surface ship handling. Current efforts on the testbed are directed towards open-water and restricted maneuvering surface ship handling.

## BACKGROUND

Many VE-based testbed systems have been developed and reported on in the virtual environment research community (Appino, Lewis, Koved, Ling, & Codella, 1992, Pausch, Burnette, Conway, DeLine, & Gossweiler, 1994, Bricken, & Coco, 1993, Shaw, C., Green, M., Liang, J., & Sun, Y. 1993, Davidson, 1996, Davidson, 1997). A review of such efforts provides a baseline in developing requirements for a rapid-prototyping, reconfigurable software testbed. Based on past research, Table 1 reveals some key features that should be considered in the development of a VE-research testbed.

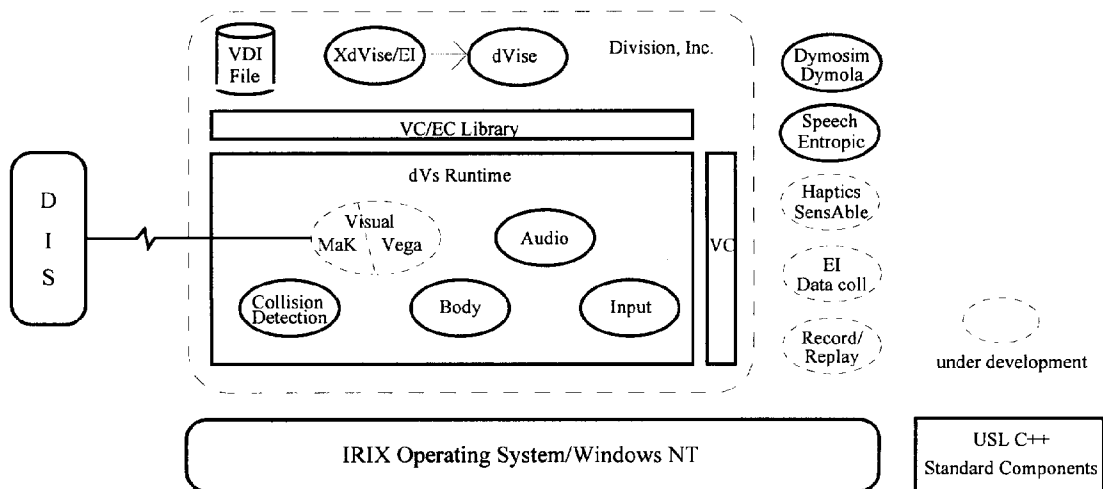
Performance	Design
SS Smoothly generated images -Database organization -Efficient processing by renderer	Performance Measures -Data collection, recording, playback and analysis of session data
Lo Low tracking latency -Communication overhead minimized -Predictive techniques implemented -Continuous sampling makes tracking data readily available	Co Reconfigurable -Database-driven models/process configuration
Low event latency -Sensory events are synchronized without noticeable delay to the user	Device Independence -Database definitions of device types/mounting/units of measure -API is free of device dependencies
Di Distributed processing -One machine/processor per task	Portable representations across heterogeneous platforms -ASCII file definitions of various representations
Ef Efficient communication -Low overhead	Extendible database tools for adding new properties to objects

Table 1. VE Testbed Performance and Design Criteria

## VETT TESTBED FUNCTIONAL COMPONENTS

MSTT's approach in creating the VETT software testbed is to use software components from Division Inc., Entropic Research, SensAble Technologies, Paradigm Simulation, and Dynasim AB to provide the needed software capability. Division Inc. provides the necessary database infrastructure, communications, and processes to create basic VE applications. Entropic Research provides tools for speech recognition. SensAble Technologies provides hardware and software to simulate sensory input involving touch. Dynasim AB specializes in object-oriented simulation and provides the needed capability to simulate symbolic systems of equations, and Paradigm Simulation provides DIS and marine effects capabilities. C++ is used for in-house process development, and UNIX Systems Lab's C++ Standard Components Library is used to provide common tools for software development.

The most difficult aspect of testbed development is the integration of other subsystems so that the use of added features is transparent to the user and software developer. Figure 1 shows all the major functional components of the VETT testbed. The simulation system, speech recognizer, haptic interface, data collection, record/replay, and augmentations to the visual renderer are all new servers that have been added to the basic VETT kernel. The following sections provide brief descriptions of the testbed functional components.



**Figure 1. VETT Testbed Software Components**

### TESTBED INFRASTRUCTURE (Division Inc.)

**dVS Runtime.** The VETT testbed infrastructure is built on the dVS Runtime/dVISE software environment from Division Inc. It provides a standard suite of "Actors" that are typically required by all VE applications. The dVS Runtime includes processes to support visual rendering, sound spatialization, collision detection, 3D position tracking, and the virtual body. Each Actor is reconfigurable, and the interface to control the various processes of the dVS Runtime is common across a number of different platforms (Division, 1996).

The dVS Runtime environment provides facilities to write user-developed applications as well as the ability to create third-party Actors which can be integrated with the dVS Runtime. Additionally, Division provides a shared-database communication scheme which is available to third-party Actors. The interprocess communications (IPC) is based on semaphores and shared memory of the native operating system. Consequently, Division's IPC is high-speed and can be used for periodic or asynchronous updates. Division also supports networked, collaborative VEs.

**dVISE.** A Virtual Data Interchange (VDI) database is used as input to Division's interactive VE authoring tool, dVISE. The VDI script is an ASCII database which is used to describe objects in the VE as well as events that occur as a result of user or object interactions. Objects can be inherited from VDI libraries and augmented with additional behaviors. The authoring/user environment supports many different 2D and 3D peripheral devices and allows the participant to create VEs interactively. Division also provides the ability to expand dVISE with user-defined events which makes it possible to tailor dVISE to meet the user's unique requirements.

## **FORCE-FEEDBACK SYSTEM (SensAble Technologies)**

The haptics capability of the VETI testbed comes from SensAble Technologies' PHANTOM haptic interface and Ghost software development kit (SDK). The Haptics Actor is designed to fit in with the other Actors that make up the dVS Runtime. The design assumes that the definition of the haptic database is external; haptics attributes, defined in the VDI database, are attached to entities in the VE. The Haptic Actors must monitor these entities and reflect their state (position etc.) within the haptic database. The Haptics Actor also monitors and reports the position of the PHANTOM input stylus to the dVS Runtime environment (Division, 1997).

## **SIMULATION SYSTEM (Dynasim AB)**

Dymola. Dymola, from Dynasim AB, is an object-oriented software tool for modeling large continuous-time systems and discrete events. The tool allows the modeler to create simple "base" classes that abstract shared physical interfaces between various components in a large system. The base classes are typically used to define what a "connection" is between two or more components (Elmqvist, 1994).

Dymosim. Dymosim is C program text that is generated from the Dymola language description. Additionally, Dynasim AB provides utilities to create a real-time version of Dymosim that contains fixed-step size integration routines, a shared-memory API for control, and access to all model parameters, inputs, outputs, and state variables via shared-memory. Configuration files are used to select the integration scheme, initialize variables, and determine the duration of the simulation for non real-time applications. Additionally, any number of processes can be attached to shared memory. This is useful for displaying the results of the simulation to the participant in the VE or for data collection.

## **VISUAL SYSTEM (Paradigm Simulation Inc.)**

Vega. In order to provide a realistic environment to conduct ship handling experiments, it was necessary to add marine effects to the testbed capability. Instead of designing a new visual Actor from scratch, it was decided to augment Division's existing visual Actor with the marine special effects of Vega Marine provided by Paradigm Simulation. Vega application definition files (ADF) are used to configure the entire Vega system by incrementally loading ADFs based on environment variables. Once the Vega system is configured, all system definitions are resolved to ensure that the loaded configuration is valid.

VR-Link. Although DIS is not required in current testbed applications, a DIS interface is now possible via the DIS option of Vega. Vega DIS is built on top of Mäk Technologies VR-Link. Details as to the full implementation are still being considered.

## **SPEECH RECOGNITION (Entropic Research)**

HAPI/HTK. Entropic Research's HAPI speech recognition system was added to the testbed as a needed capability to command various controls of an application using voice. HAPI provides all the features of a modern speech recognizer including acoustic speech models based on Hidden Markov Models, pronunciation dictionary, language models, and GUI-based tools for constructing application grammars (Odell, Ollason, Valtchev, & Whitehouse, 1997)

The Speech Recognition Actor is classed as an Input Actor, taking as input vocal utterances and converting them into sequences of key-press events that are reported using the conventional dVS input stream. This input stream is monitored by other Actors in the dVS Runtime, including dVISE. Currently, only tagged ASCII characters are used as input to dVISE from the Speech Actor. Future plans for natural language processing are being considered.

## **DATA COLLECTION/RECORD/REPLAY**

Data Collection. The Data Collection Actor is designed as a configurable Actor with interfaces to the dVS runtime environment and the Dymosim shared-memory arena. Data collection is broken out into information that is periodic and aperiodic. Periodic data comes from two different sources: 1) the simulation, and 2) the participant's body movements. Aperiodic data is generated primarily from the participant's interactions in the VE via dVISE.

Record/Replay. As a means of data elicitation and subject feedback, a Record/Replay Actor is currently under development. The ability to record a session with playback is complete. Future plans include first or third person review, fast forward/reverse, pause, and the ability to step into the simulation during replay.

## REFERENCES

- Appino, P., Lewis, J. B., Koved, L., Ling, D. T., & Codella, C. F. (1992). An Architecture for Virtual Worlds. Presence: Teleoperators and Virtual Environments, 1,1, pp. 1-17.
- Bricken, W., & Coco, G. (1993). The VEOS Project. Tech. Rep. No. R-93-3. Seattle: Human Interface Technology Laboratory.
- Davidson, S. W. (1996). Software Design of a Virtual Environment Training Technology Testbed and Virtual Electronic System Trainer (NAWCTSD Tech Rep. TR96-002). Orlando, FL: Naval Air Warfare Center Training Systems Division.
- Davidson, S. W. (1997). Design of an Open-Water Ship Handling Software Testbed (NAWCTSD Tech Rep. TR97-007). Orlando, FL: Naval Air Warfare Center Training Systems Division.
- Division, Inc. (1996). dVISE for Unix Workstations, User Guide. Division Inc., San Mateo, CA.
- Division, Inc. (1997). Report on Haptic Actor for VE Testbed and dVS Version 1. Division, Inc. San Mateo, CA.
- Elmqvist, H. (1994). DYMOLA - Dynamic Modeling Language User's Manual. Dynasim AB, Lund Sweden.
- Odell, J., Ollason, D., Valtchev, V., Whitehouse, D. (1997). The HAPI Book, A description of the HTK Application Programming Interface, Version 1.0. Entropic Cambridge Research Laboratory, LTD.
- Pausch, R., Burnette, T., Conway, M., DeLinc, R., & Gossweiler, R. (1994). Alice: A Rapid Prototyping System for Virtual Reality. Course Notes 17, ACM SIGGRAPH 1994. pp. 5:1 - 5:6.
- Shaw, C., Green, M., Liang, J., & Sun, Y. (1993). Decoupled Simulation in Virtual Reality with The MRTToolkit. ACM Transactions on Information Systems, Vol 11, No. 3, pp. 287-317.